2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW) | 979-8-3503-8403-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDEW61823.2024.00052

Integrating Complex Pangenome Graphs

Jérôme Arnoux[1] Angela Bonifati[2] Alexandra Calteau[1] Stefania Dumbrava[3] Guillaume Gautreau[4]

[1]*Genoscope/LABGeM - CEA, CNRS, Paris Saclay University* [2]*IUF, CNRS LIRIS, Lyon 1 University*
[3]*SAMOVAR/Inst. Poltech de Paris, ENSIIE* [4]*MaIAGE, Université Paris-Saclay, INRAE*

***Abstract*—Graph databases are increasingly used to handle complex data pipelines, in which interconnected data is exploited for visualization and analytics. We propose a novel method, PanGraph-DB, for performing complex inter-pangenomic analysis within a graph database. As a case study, we focus on the antibiotic resistance in sequenced genomes. Over the past decade, the volumes of genomic data stored in public databases have grown exponentially, to the point of hindering comparative genomics algorithms. We show that, due to the nature of genomic data, graph databases enable accurate data and metadata analysis, visualization, and comparison across diverse genomes in the pangenomic approach. Families of graph-encoded pangenomes can then be integrated under a common mediated graph schema. The graph data integration allows to visualize and compare several pangenomes, as well as to analyze AntiMicrobial Resistance (AMR) gene niches through a combination of graph queries, whose performance and scalability we study.**

I. Introduction

Graphs are ubiquitous in several applications that rely on interconnected data to represent, explore, predict, and explain real- and digital-world phenomena. In the near future, graph ecosystems are expected to handle complex data pipelines, ranging from data pre-processing, querying, and analysis, to advanced processing, through learning and inference [1]. To optimize for performance and accuracy, such complex data pipelines need to be purposed for the particular tasks they target. In this paper, we focus on devising a custom methodology for enabling comparative genomics on pangenome graphs.

Typical analyses in comparative genomics often rely on a reference-centric approach to grasp species diversity, based only on several genomes. This reference genome, however, fails to provide sufficient coverage. A trivial solution would be to pairwise compare all the known genomes, but this would lead to a combinatorial explosion. The pangenomic approach overrides these limitations, by combining all the genomes, including the reference ones, in a unified data structure. This can be represented using various formalisms, *e.g.*, sets, Multiple Sequence Alignments, Sequence graphs, and De Bruijn graphs, as reviewed in [2]. Among these, microbial pangenome graphs increasingly rely on nodes, corresponding to clusters of similar genes (families), linked by edges, indicating their genomic neighborhood in various genomes [3], [4], [5], [6].

An open problem in pangenomics is how to compare several pangenome graphs straightforwardly. In particular, deciphering transfers of genetic information between species (pangenomes), such as AMR genes, raises many critical issues. The first hurdle is the size of the combined graphs, of the order of millions of nodes, requiring custom solutions for storage and efficient computation. Second, querying the graphs, to find similar modules for instance, can be difficult, in terms of both algorithmic and computational complexity.

We address these challenges in a practical system, by importing pangenome families in a unified property graph, under a mediated schema. The schema helps domain experts understand and explore the multi-pangenome graph and formulate graph queries that facilitate complex bioinformatics tasks, such as AMR analyses. Our method leverages the Neo4j system [7] and our queries are expressed in its native openCypher [8] language. These can, however, be equivalently encoded in any graph query language, including the future GQL [9] standard. We establish the *scalability* of the approach when varying the number of pangenome graphs, and its *efficiency* on custom AMR queries. Overall, *our work shows how to solve a complex domain-specific task, which has been considered unfeasible in classical genomics, by designing a dedicated graph processing pipeline*. To the best of our knowledge, *ours is the first work that uses graph databases for the real-world use-case of efficient and scalable multi-pangenome processing*. Our promising first results, obtained in the context of investigating the AMR of various pangenomes, open the perspective of employing this methodology in further applications. We make our PanGraph-DB artifact [10] publicly available.

Related Work. Graph databases have gained rapid adoption in the life sciences, as surveyed in [11] and as witnessed by creation of various datasets, *i.e.*, BioRDF for linked open data, GeneOntology for gene taxonomies, GProfile - for metabolism information, KEGG for gene and genome information, ChEBI for chemical entities, and the Genomic Data Model [12] for omics data. PanTools [13] is the work closest to ours, as it also uses the Neo4j graph database for comparative genomics. Their technique, though, hinges on a De Bruijn Graph and facilitates the integration of eukaryotic pangenomes. Compared to our approach, De Bruijn Graphs are at a lower level of granularity, which is subject to high variability and makes it challenging to scalably interpret and analyze functional and structural patterns across microbial species.

Tertiary data analysis has been carried out with Genomic Data Model (GDM) and GenoMetric Query Language (GMQL), to enable scientists and bioinformaticians to focus on the biological questions and the design of their experimental studies, instead of implementing the computational pipelines across different formats [12]. They focus, however, on genomic regions, and their comparisons are implemented as joins in the GMQL queries. Their query language is not

979-8-3503-8403-1/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDEW61823.2024.00052

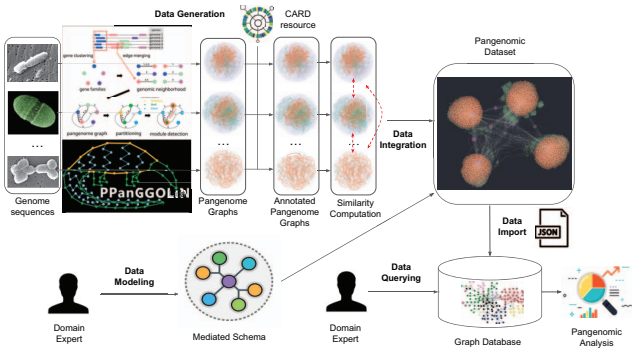

Fig. 1: Graph database driven pangenomic analysis.

graph-oriented and is thus not applicable to comparisons of families of pangenomic graphs and AMR identification.

II. Methodology

Our methodology aims to facilitate efficient comparative pangenomics. As such, we constructed the PanGraph-DB system, whose pipeline (see Figure 1) is capable of operating on pangenome graphs, computed by the PPanGGOLiN [5] framework, and of leveraging graph databases for integrated analyses by domain experts. Our approach is *system agnostic* and can be reproduced with any graph database whose data model and querying capabilities are comparable to those of the Neo4j system we employ. Given the rich *property graph data model* supported by Neo4j, domain experts can further enrich such pangenome graphs with custom properties. We illustrate this technique with a complex analysis aimed at inspecting AMR patterns in a multi-pangenome setting. Domain experts can, however, adapt and extend it to various other applications.

Data generation. Our pipeline (see Figure 1) takes as input complete ESKAPE genomes [14], from the NCBI GenBank database [15]. These are then processed by PPanGGOLiN, which performs gene clustering, edge merging, and statistical partitioning to compute pangenome graphs. Using these, we can highlight Regions of Genomic Plasticity (RGP) that are relevant to our analyses and that correspond to genomic spots (hotspots) onto which AMR genes can be integrated. We compute the information regarding *RGPs* using PPanGGOLiN's PanRGP method [16] and we further connect co-occurring and co-located gene families. Indeed, these might potentially be involved in a common biological process, as is commonly the case for AMR genes. Hence, we regroup them into structures called *modules*, with PPanGGOLiN's panModule method [17].

Gene families from these pangenome graphs are enriched with Comprehensive Antibiotic Resistance Database (CARD) annotations, in order to identify known AMR genes [18]. Also, various similarity levels between gene families of different pangenomes are computed, through dedicated alignment methods [19]. Finally, we obtain multiple CARD-annotated pangenome graphs, incorporating additional information regarding the partitions, RGPs, and modules the genomes relate to. This information, however, is not explicit and cannot be directly queried. Moreover, the graphs are largely disconnected, except for the similarity annotations between gene families.

Data modeling. As genomic analyses typically require a holistic view, encompassing information stored in all of these individual datasets, it is important to integrate them into a single graph. This is especially challenging, as the pangenome data lacks structural information and explicit labeling. To address this, we first construct a unifying schema that will shape the integrated multi-pangenome instance to be imported and analyzed in the Neo4j database. A key requirement for inter-pangenome analyses is having a data model that is expressive enough to capture multi-pangenome properties. Hence, for our AMR task, we need to enrich the previously computed datasets with further metadata, as follows. Each pangenome corresponds to a particular species, carries a mandatory name and unique identifier, and is linked to all the gene families it comprises. Note that all families must be associated with exactly one partition (persistent, shell, or cloud). To model the possible connections between families, we need to first determine whether they are part of the same pangenome. Intra-pangenome families can be marked as neighbors and, based on this, neighborhood weights can be computed, by counting their number of genomes. Inter-pangenome families can only be linked through similarity relations that can be characterized by a percentage of identity and a percentage of coverage. To facilitate the analyses, for each module, we explicitly store information regarding its gene families. Next, for each gene, we record its name, its *start* and *stop* position on the DNA sequence (*contig*), as well as its *RGPs*, which we connect to named *spots*, if they are co-located in the pangenome graph.

Data processing. The data import methodology closely follows that of the CovidGraph framework [20]. First, we create a Python dictionary for every pangenome. This has a hierarchical structure, whose parent is the pangenome itself and whose leaves are the genes. As such, we can use the *dict2graph* package [21] to create relationships between nodes, load properties for nodes and relations, as well as automatically index and merge all nodes and relationships into a graph. Note that all these tasks are parallelized. Next, we sequentially load the similarities of edges, by inspecting the alignment result table and extracting pairs of families with identity and coverage greater than 30% and 80%. Finally, we create edges between family nodes with the graphio package [22]. The expert user can then visualize, explore, and inspect the data through graph queries that can extract complex patterns [23].

III. Integrated Pangenomics

To efficiently perform genomic analyses on our graph dataset of connected pangenomes, we leverage the Neo4j *graph database*. This natively stores data as a *property graph*, *i.e.*, a directed, multi-labeled multi-graph with key/value properties attached to nodes and edges. To facilitate data integration, we design a *mediated schema* (Figure 2) that captures the integrated dataset structure. `Pangenome` nodes are connected to their genomic `Family` nodes and to neighboring and similar nodes of the same label in `Modules`. `Family`

Fig. 2: Mediated Schema.

nodes contain antibiotic resistance annotations. These nodes are classified as belonging to persistent, shell, and cloud `Partitions`, depending on whether their `Genes` are prevalent in all, some, or only a few corresponding genomes. Finally, `Genes` are part of `Contigs`, associated to particular `Genomes`, and can be part of `RGPs`, located in `Spots`.

Antibiotic Resistance Analysis. We illustrate the utility of graph databases for AMR identification, a key genomics task. Grasping the evolution of such resistance profiles across pangenomes is crucial to understanding how these elements spread between species. To conduct the analysis, the domain expert has to first comprehend the scale of such AMR profiles within the various pangenomes. She can extract aggregated information from the graph and identify the pangenomes with the highest number of relevant biological features at the intra-pangenomic level. Thus, she can express top-k queries [10] to focus on information about pangenomes (Q1, Q2, Q4), RGPs and Spots (Q3), as well as modules (Q5) that contain the most CARD-annotated families and top-k species in terms of modules (Q6). These queries allow gaining AMR insights about a species. However, to determine evolutionary patterns, one needs to analyze multiple pangenomes. Hence, a further step is to consider pangenome pairs and extract families that have *similar* AMR profiles. This can be encoded (Q7) through a main query that recovers, for a pangenome, its families and their modules, and two correlated sub-queries that further extract, for each previously identified family, all other similar families, from other pangenomes. Note that one has to explicitly export the variables reused in the correlated queries.

Q7. *Return the names of similar inter-pangenomic families and of the partitions and pangenomes they belong to.*

```
MATCH (p1:Pangenome)<-[:IS_IN_PANGENOME]-
(f1:Family)-[:HAS_PARTITION]->(s1:Partition)
WITH p1, f1, s1
MATCH (f1)-[:IS_SIMILAR]-(f2:Family)-[:
    HAS_PARTITION]->(s2)
WITH p1, f1, s1, s2, f2
MATCH (p2:Pangenome)<-[:IS_IN_PANGENOME]-(f2)
WHERE p1.name <> p2.name
RETURN p1.name, f1.name, s1.partition,
p2.name, f2.name, s2.partition
```

The query below is key to the analysis, as the information regarding `Spot` nodes allows highlighting RGP hotspots, *i.e.*, common insertion sites for Horizontal Gene Transfers.

Q8. *Identify similar inter-pangenomic AMR families where at least one has RGP-related genes, part of hotspots.*

Note that in the Cypher encoding, we explicitly name the graph patterns to filter intermediate results iteratively, using path-level reachability constraints. As such, we first specify the pattern allowing to identify families that have RGP-related genes and extract the graph objects connected to these, in particular the `Spot` information. We then refine the results with a path condition aimed at identifying pairs of similar families, within different pangenomes, and extract further information regarding their `Partition`. We also discard results wherein families do have not any AMR annotation.

```
MATCH
a=(p:Partition)<-[HAS_PARTITION]-(f1:Family)
-[:IS_IN_FAMILY]-(g:Gene)-[:IS_IN_RGP]-(r:RGP)
-[:IS_IN_SPOT]-(s:Spot)
WITH a,p,f1,g,r,s
MATCH b=(p1:Pangenome)-[:IS_IN_PANGENOME]-(f1)
-[:IS_SIMILAR]-(f2:Family)
-[:IS_IN_PANGENOME]-(p2:Pangenome)
WHERE f1.annotation IS NOT NULL AND z1<>z2
WITH a,b,p,f1,g,r,s,p1,zp2,f2
MATCH c=(f2)-[:HAS_PARTITION]-(p2)
RETURN a,b,c
```

Next, we can precisely compare the neighborhoods (contexts) of modules across pangenomes with the below query.

Q9. *Return the names and count of similar families from different pangenomes and the names of their modules.*

```
MATCH (m1:Module)<-[:IS_IN_MODULE]-(f1:Family)
      -[:IS_IN_PANGENOME]->(p1:Pangenome)
WITH f1, m1, p1
MATCH (p2:Pangenome)<-[:IS_IN_PANGENOME]-(f2:
    Family)-[:IS_SIMILAR]-(f1)
WITH f1, m1, p1, f2, p2
MATCH (f2)-[:IS_IN_MODULE]-(m2:Module)
WHERE f1.annotation IS NOT NULL AND p1 <> p2
RETURN p1, p2, m1, m2, f1, f2
```

Interestingly, families with similar CARD annotations may even belong to different partitions, highlighting the genetic variability between species, as analyzed with the query below.

Q10. *Return the names of the top 10 pairs of similar families, whose identity and coverage metrics surpass the 0.8 threshold, their partition names, and their CARD annotations.*

```
MATCH
(f1:Family)-[:HAS_PARTITION]->(s1:Partition)
WITH f1, s1
MATCH (f1)-[r1:IS_SIMILAR]->(f2)
      -[:HAS_PARTITION]->(s2:Partition)
WHERE r1.identity >= 0.8
AND r1.coverage >= 0.8
RETURN f1.name, s1.partition, f2.name,
s2.partition, r1.identity, r1.coverage,
f1.annotation, f2.annotation
ORDER BY r1.identity DESC LIMIT 10
```

Analyzing the query results, very similar inter-pangenomic families could indicate recent inter-species gene transfers.

IV. Experimental Evaluation

We performed scalability experiments on a virtual machine, running Ubuntu (version 22.04.1 LTS), equipped with an Intel Xeon Processor (Skylake, IBRS), clocked at 2.2Ghz, 153GB of RAM, and 621GB of free hard drive space. The queries were executed in the Neo4j Community Edition (version 4.4.12) and the dataset characteristics are available in our PanGraph-DB artifact [10]. Note that data import is one of the bottlenecks faced when processing multiple pangenomes. In previous analyses, even the simultaneous data loading of several PPanGGOLiN files (one HDF5 file per pangenome) was challenging in terms of memory usage. We deem our runtimes acceptable for massive cross-pangenome analyses, as they require less than a workday to fully import the datasets. The disk usage is also sustainable, given the compactness of storing 10 pangenomes (6.8GB), corresponding to billions of genes, thousands of families and genomes, as well as expert biological information (RGPs, Spots, AMR annotations).

Quantitative Analysis. We assess the *performance* and *scalability* of our methodology on the previous complex pangenomic analyses. As such, we analyze the runtimes of evaluating queries Q1-Q10 on pangenome datasets integrating an increasing number of pangenomes (see Figure 3). We note that the query execution times range from approx. 27.07 ms. on average (Q2) to approx. 66.8 ms. on average (Q8), when considering only two pangenomes and from approx. 31.8 ms. (Q5) to approx. 88.93 ms. (Q7), when considering ten pangenomes. The minimal execution times for Q2 and Q5 can be explained by the relative simplicity of the queries, as this computes count aggregates over a basic path comprised of only two edges. The maximal execution times are recorded for queries Q7 and Q8. Both are complex-correlated queries containing several subqueries. In terms of scalability, we note that the increase in execution time is nearly linear or sublinear when progressively adding more pangenomes. Moreover, we can see that the most complex queries, *i.e.*, Q3, Q7, and Q10, which also take longest to execute, exhibit the highest variability, as witnessed by their observed standard deviation. This indicates that performances start to deteriorate when considerably increasing data volumes, as the dataset integrating ten pangenomes records performance variations of up to 40%. Dealing with such scenarios requires further optimizations that graph processing systems are expected to support in the future. Nevertheless, the pangenome sizes we consider are already well beyond what can be supported by current methods that, moreover, do not support inter-pangenome analyses at all.

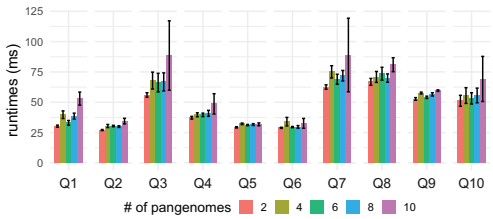

Fig. 3: Average Q1-Q10 runtimes on multiple pangenomes.

Qualitative Analysis. While resistance islands have been extensively studied in the microbiology literature [24], [25], [26], to the best of our knowledge, *ours is the first data-driven pangenomic approach to identify and compare them at scale*. Henceforth, we evaluate Q9 (Section III) on our largest dataset (ten pangenomes) and explore the results. Note that Q9 is relevant for the entire AMR pipeline, as it helps to identify similar inter-pangenome modules that contain AMR annotated genes. RGPs carrying such AMR modules may correspond to resistance islands, i.e., mobile genetic elements that contain multiple resistance genes and may be passed between species (pangenomes) through Horizontal Gene Transfers. It is thus particularly important to expose these genomic patterns to better understand their spread in ESKAPE bacteria.

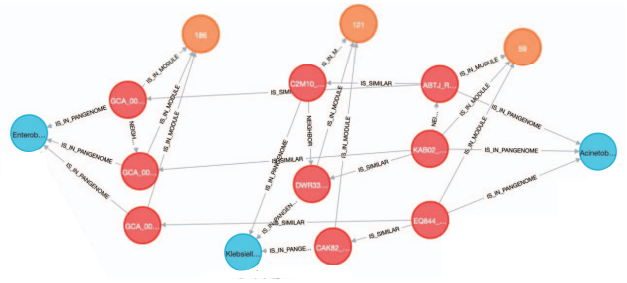

Fig. 4: Similar modules (orange) with AMR-related families (red) in distinct pangenomes (blue).

Inspecting the result graph, we can identify and extract relevant patterns. For example, between the *E. kobei*, *K. pneumoniae* and *A. baumanii* pangenomes (in blue), there are three pairs of similar gene families (in red) that are part of similar modules (in orange) in between each of the three species (see Figure 4). Having isolated these three modules, the expert user can inspect their membership in relevant resistance islands.

We notice that module identification is greatly facilitated by using a graph database, as all such pairs can be extracted with a single, declarative query. Previously, no such direct methodology for inter-pangenomic analysis and exploration was readily available. This qualitative study also revealed the importance of the underlying data model, which we will extend to explicitly represent resistance island conglomerates.

V. Conclusion and Perspectives

We have introduced the novel PanGraph-DB framework and showcased the usage of a graph database for complex pangenomic processes. By defining a mediated schema on top of several isolated families of pangenomes, we have created a unified pangenome graph that can be inspected for both intra- and inter-pangenomic analyses. We have also experimentally established the feasibility of data loading, processing, and querying with our method. Our approach is generic and can be deployed in other graph databases, as we make PanGraph-DB readily available to both the database and bioinformatics communities. We intend to optimize our pipeline to support even larger pangenomes and to extend our data model with other interesting types of genomic data, *e.g.*, metabolic pathways, defense, and virulence islands.

REFERENCES

[1] S. Sakr, A. Bonifati, H. Voigt, A. Iosup, K. Ammar, R. Angles, W. G. Aref, M. Arenas, M. Besta, P. A. Boncz, K. Daudjee, E. D. Valle, S. Dumbrava, O. Hartig, B. Haslhofer, T. Hegeman, J. Hidders, K. Hose, A. Iamnitchi, V. Kalavri, H. Kapp, W. Martens, M. T. Özsu, E. Peukert, S. Plantikow, M. Ragab, M. Ripeanu, S. Salihoglu, C. Schulz, P. Selmer, J. F. Sequeda, J. Shinavier, G. Szárnyas, R. Tommasini, A. Tumeo, A. Uta, A. L. Varbanescu, H. Wu, N. Yakovets, D. Yan, and E. Yoneki, "The future is big graphs: a community view on graph processing systems," *Communications of the ACM*, vol. 64, no. 9, pp. 62–71, 2021.

[2] Computational PanGenomics Consortium, "Computational pan-genomics: status, promises and challenges," *Brief. Bioinform.*, vol. 19, no. 1, pp. 118–135, Jan. 2018.

[3] Y. Peng, S. Tang, D. Wang, H. Zhong, H. Jia, X. Cai, Z. Zhang, M. Xiao, H. Yang, J. Wang, K. Kristiansen, X. Xu, and J. Li, "MetaPGN: a pipeline for construction and graphical visualization of annotated pangenome networks," *Gigascience*, vol. 7, no. 11, Nov. 2018.

[4] S. C. Bayliss, H. A. Thorpe, N. M. Coyle, S. K. Sheppard, and E. J. Feil, "PIRATE: A fast and scalable pangenomics toolbox for clustering diverged orthologues in bacteria," *Gigascience*, vol. 8, no. 10, Oct. 2019.

[5] G. Gautreau, A. Bazin, M. Gachet, R. Planel, L. Burlot, M. Dubois, A. Perrin, C. Médigue, A. Calteau, S. Cruveiller, C. Matias, C. Ambroise, E. P. C. Rocha, and D. Vallenet, "Ppanggolin: Depicting microbial diversity via a partitioned pangenome graph," *PLoS Comput. Biol.*, vol. 16, no. 3, 2020.

[6] G. Tonkin-Hill, N. MacAlasdair, C. Ruis, A. Weimann, G. Horesh, J. A. Lees, R. A. Gladstone, S. Lo, C. Beaudoin, R. A. Floto, S. D. W. Frost, J. Corander, S. D. Bentley, and J. Parkhill, "Producing polished prokaryotic pangenomes with the panaroo pipeline," *Genome Biology*, vol. 21, no. 1, p. 180, Jul. 2020.

[7] Neo4j, *Neo4j Graph Database*, Std., 2023. [Online]. Available: http://neo4j.org/

[8] ——, *OpenCypher*, Std., 2023. [Online]. Available: http://opencypher.org/

[9] GQL Standards Committee, *GQL*, Std., 2023. [Online]. Available: https://www.gqlstandards.org/

[10] (2023) PanGraph-DB. [Online]. Available: https://github.com/jpjarnoux/PanGraph-DB

[11] S. Timón-Reina, M. Rincón, and R. Martínez-Tomás, "An overview of graph databases and their applications in the biomedical domain," *Database J. Biol. Databases Curation*, 2021.

[12] M. Masseroli, A. Canakoglu, P. Pinoli, A. Kaitoua, A. Gulino, O. Horlova, L. Nanni, A. Bernasconi, S. Perna, E. Stamoulakatou, and S. Ceri, "Processing of big heterogeneous genomic datasets for tertiary analysis of next generation sequencing data," *Bioinformatics*, vol. 35, no. 5, pp. 729–736, 2019.

[13] E. M. Jonkheer, D.-J. M. van Workum, S. Sheikhizadeh Anari, B. Brankovics, J. R. de Haan, L. Berke, T. A. J. van der Lee, D. de Ridder, and S. Smit, "PanTools v3: functional annotation, classification and phylogenomics," *Bioinformatics*, vol. 38, no. 18, pp. 4403–4405, 07 2022. [Online]. Available: https://doi.org/10.1093/bioinformatics/btac506

[14] M. S. Mulani, E. E. Kamble, S. N. Kumkar, M. S. Tawre, and K. R. Pardesi, "Emerging strategies to combat eskape pathogens in the era of antimicrobial resistance: A review," *Frontiers in Microbiology*, vol. 10, pp. 1–24, 2019.

[15] E. W. Sayers, J. Beck, E. E. Bolton, D. Bourexis, J. R. Brister, K. Canese, D. C. Comeau, K. Funk, S. Kim, W. Klimke, A. Marchler-Bauer, M. Landrum, S. Lathrop, Z. Lu, T. L. Madden, N. O'Leary, L. Phan, S. H. Rangwala, V. A. Schneider, Y. Skripchenko, J. Wang, J. Ye, B. W. Trawick, K. D. Pruitt, and S. T. Sherry, "Database resources of the national center for biotechnology information," *Nucleic Acids Res.*, vol. 49, no. D1, pp. D10–D17, Jan. 2021.

[16] A. Bazin, G. Gautreau, C. Médigue, D. Vallenet, and A. Calteau, "panRGP: a pangenome-based method to predict genomic islands and explore their diversity," *Bioinformatics*, vol. 36, no. Suppl_2, pp. i651–i658, Dec. 2020.

[17] A. Bazin, C. Medigue, D. Vallenet, and A. Calteau, "panmodule: detecting conserved modules in the variable regions of a pangenome graph," *bioRxiv*, 2021. [Online]. Available: https://www.biorxiv.org/content/early/2021/12/07/2021.12.06.471380

[18] B. P. Alcock, A. R. Raphenya, T. T. Y. Lau, K. K. Tsang, M. Bouchard, A. Edalatmand, W. Huynh, A.-L. V. Nguyen, A. A. Cheng, S. Liu, S. Y. Min, A. Miroshnichenko, H.-K. Tran, R. E. Werfalli, J. A. Nasir, M. Oloni, D. J. Speicher, A. Florescu, B. Singh, M. Faltyn, A. Hernandez-Koutoucheva, A. N. Sharma, E. Bordeleau, A. C. Pawlowski, H. L. Zubyk, D. Dooley, E. Griffiths, F. Maguire, G. L. Winsor, R. G. Beiko, F. S. L. Brinkman, W. W. L. Hsiao, G. V. Domselaar, and A. G. McArthur, "CARD 2020: antibiotic resistome surveillance with the comprehensive antibiotic resistance database," *Nucleic Acids Res.*, vol. 48, no. D1, pp. D517–D525, Jan. 2020.

[19] M. Steinegger and J. Söding, "MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets," *Nature Biotechnology*, vol. 35, no. 11, pp. 1026–1028, Nov. 2017.

[20] HealthECCO, "Covidgraph," https://covidgraph.org/ (visited: 07-11-2022), 2021.

[21] T. Bleimehl. (2023) dict2graph. [Online]. Available: https://git.connect.dzd-ev.de/dzdpythonmodules/dict2graph

[22] M. Preusse. (2023) Graphio. [Online]. Available: https://graphio.readthedocs.io/en/latest/

[23] A. Bonifati and S. Dumbrava, "Graph queries: From theory to practice," *SIGMOD Rec.*, vol. 47, no. 4, pp. 5–16, 2018.

[24] J. Hacker and J. B. Kaper, "Pathogenicity islands and the evolution of microbes," *The Annual Review of Microbiology*, vol. 54, pp. 641–679, 2000.

[25] O. Gal-Mor and B. B. Finlay, "Pathogenicity islands: a molecular toolbox for bacterial virulence," *Cellular Microbiology*, vol. 8, no. 11, pp. 1707–1719, Nov 2006.

[26] S. Algarni, S. C. Ricke, S. L. Foley, and J. Han, "The Dynamics of the Antimicrobial Resistance Mobilome of Salmonella enterica and Related Enteric Bacteria," *Frontiers in Microbiology*, vol. 13, p. 859854, 2022.